



Digitized by the Internet Archive
in 2013

<http://archive.org/details/onorderingenumer850troj>

62
850

Math

8

UIUCDCS-R-77-850

UIIU-ENG 77 1706

ON THE ORDERING, ENUMERATION AND
RANKING OF k -ARY TREES

by
Anthony E. Trojanowski

February, 1977

the Library of the
APR 12 1977
University of Illinois
Champaign



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

UIUCDCS-R-77-850

On the Ordering, Enumeration and
Ranking of k -ary Trees*

Anthony E. Trojanowski

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801

February 1977

* This work was supported in part by the National Science Foundation under Grant Numbers DCR 74-02774 and MCS 73-03408,

§I. Introduction: The main problem of classical combinatorics may be described as follows: given a finite set X , evaluate $N = |X|$. Evaluation of N implicitly defines $N!$ set isomorphisms

$$\varphi: X \rightarrow \{0, 1, \dots, N-1\}.$$

Any choice of such a φ induces a total order $<_{\varphi}$ on X ,

$$x <_{\varphi} y \text{ iff } \varphi(x) < \varphi(y).$$

φ is thus an order isomorphism between the ordered sets $(X, <_{\varphi})$ and $\{0, 1, \dots, N-1\}$ with the usual ordering.

With X and N as above, a somewhat different problem may be defined:

impose a total order $<_X$ on X and construct an order isomorphism

$$\varphi: (X, <_X) \rightarrow \{0, 1, \dots, N-1\}.$$

Such a function is a ranking function on X with respect to $<_X$. Clearly φ is unique, and also, for $x \in X$

$$\varphi(x) = |\{y \in X: y <_X x\}|.$$

A converse problem may be defined: given $X, <_X, N$ as above, compute φ^{-1} .

In this paper, X will be the set of k -ary trees on n vertices,

$k \geq 2, n \geq 0$.

1.1 Definition [k-ary tree]: Let k be an integer, $k \geq 2$.

(i) ϕ , the empty tree, is a k -ary tree with no root and with vertex

$$\text{set } V(\phi) = \phi;$$

(ii) if $T_1, \dots, T_k$ are k -ary trees, then structure produced by joining a new root r to the root of each non-empty T_j , $1 \leq j \leq k$, is a k -ary tree with root r and vertex set $\{r\} \cup V(T_1) \cup \dots \cup V(T_k)$;

(iii) there are no other k -ary trees.

1.2 Definition [Subtree rooted at v]: Let T be a non-empty k -ary tree with $T_1, \dots, T_k$ as in Definition 1.1 (ii). Let $v \in V(t)$. (i) If $v = r$, the subtree of T rooted at v is T ; (ii) Otherwise, $v \in V(T_j)$ for some j , $1 \leq j \leq k$, and the subtree of T rooted at v is the subtree of T_j rooted at v .

1.3 Definition [Parent, child, etc.]: Let T be a non-empty k -ary tree, $v \in V(T)$, T_v the subtree of T rooted at v , with $T_{v1}, \dots, T_{vk}$ as in Definition 1.1 (ii). The root of each non-empty T_{vj} is child of v , and v is the parent of such a root. $w \in T_v$ is a descendant of v , and v is an ancestor of w . If $w \neq v$, the descendant and ancestor relations are proper. A vertex with no proper descendants is a leaf. A vertex which is not a leaf is internal.

1.4 Definition [Isomorphism]: Let T_1, T_2 be k -ary trees.

- (i) If $T_1 = T_2 = \phi$, T_1 and T_2 are isomorphic.
- (ii) If T_1 and T_2 are non-empty and $T_{11}, \dots, T_{1k}, T_{21}, \dots, T_{2k}$ are as in Definition 1.1 (ii), T_1 and T_2 are isomorphic iff T_{1j} and T_{2j} are isomorphic, $1 \leq j \leq k$.

1.5 Definition [Full k -ary tree]: A k -ary tree T is a full k -ary tree iff

- (i) T is non-empty; and
- (ii) each internal vertex of T has exactly k children.

There is a relationship between k -ary trees and full k -ary trees given by the following lemma.

1.6 Lemma: The set of k -ary trees and the set of full k -ary trees are in 1-1 correspondence.

Proof: The correspondence is defined as follows:

- (i) ϕ corresponds to the full k -ary tree with one vertex;
- (ii) A non empty k -ary tree T corresponds to the full k -ary tree T' obtained from T by adding leaves to T so that every vertex of T has exactly k children in T' .

It is clear that this correspondence is 1-1.

q. e. d

This correspondence is an obvious extension of the special case $k = 2$ which is presented on p. 559 of [K].

1.7 Definition [m-Inorder Labeling]: Let T be a non-empty k -ary tree with root r ; for $v \in V(T)$, let T_v be the subtree of T rooted at v and let $T_{v1}, \dots, T_{vk}$ be as in Definition 1.1 (ii). If $T_{vj} \neq \phi$, let r_j be the root of T_{vj} . Let $0 \leq m \leq k$. The following algorithm labels T in m -inorder.

begin

procedure *inorder* (v, m);

begin

for $j := 1$ until m do

if $T_{vj} \neq \phi$ then *inorder* (r_j, m);

label [v]: $= i := i + 1$;

for $j := m + 1$ until k do

if $T_{vj} = \phi$ then *inorder* (r_j, m);

end *inorder* (v, m);

$i := 0$;

inorder (*r,m*);

end.

A search which visits the vertices of T in the same order in which they are numbered by an m -inorder labeling is an m -inorder search.

The definition of m -inorder labeling may be extended to arbitrary ordered trees, but this extension is not necessary in this paper and so the present definition is framed in terms of k -ary trees.

The cases $m = 0$ and $m = k$ are the usual pre-order and post-order labelings of T respectively; the case $k = 2$ and $m = 1$ is the usual in-order labeling of a binary tree. See [A] for standard definitions of these labelings.

Finally, to formalize the remarks at the beginning of this paper, the next definition is made.

1.8 Definition [Ranking Function]: Let $(X, <)$ be a finite, non-empty ordered set with $|X| = N$. The unique order isomorphism $\varphi: (X, <) \rightarrow \{0, 1, \dots, N - 1\}$ is the ranking function for X with respect to the total order $<$.

§II. Binary Trees; Ranking by Pre-Order Search: A binary tree is, of course, a k-ary tree with $k = 2$.

2.1 Definition [$\tilde{B}(n)$, $B(n)$]: Let n be a non-negative integer. Then

$$\tilde{B}(n) = \{\text{binary trees with } n \text{ vertices}\};$$

and

$$B(n) = |\tilde{B}(n)|.$$

Clearly $B(0) = B(1) = 1$. For $n \geq 2$, we have

$$B(n) = \sum_{v=0}^{n-1} B(v) B(n-v-1); \quad (2.2)$$

this follows Definitions 1.1 and 1.4. A generating function argument may be applied to Equation 2.2 to obtain

$$B(n) = \frac{1}{n+1} \binom{2n}{n}; \quad (2.3)$$

see §2.3.4.4 of [K].

A total order on the set of all binary trees may be defined as follows:

2.4 Definition [Pre-order Ordering]: Let T_1 and T_2 be binary trees. Then

$T_1 < T_2$ iff

(i) $T_1 \in \tilde{B}(n_1)$, $T_2 \in \tilde{B}(n_2)$ with $n_1 < n_2$, or

(ii) $T_1, T_2 \in \tilde{B}(n)$ and, defining T_{11} , T_{12} and T_{21} , T_{22} according to

Definition 1.1 (ii), $T_{11} < T_{21}$ or $T_{11} = T_{21}$ and $T_{12} < T_{22}$.

2.5 Lemma: Definition 2.4 defines a total ordering on the set of all binary trees.

Proof: Clearly, it suffices to show that Definition 2.4 defines a total ordering on $B(n)$ for each n . But this is certainly the case for $n = 0$,

1; and for $n \geq 2$ the result follows by induction on n , using Definition 2.4 (ii), and noting that $n_1 + n_2 = n - 1$. q.e.d.

The way is now clear to define the ranking function on $B(n)$ with respect to the pre-order ordering.

2.6 Definition [Pre-order ranking on $B(n)$]: Let $T \in B(n)$. If $n \geq 1$ define

T_1 and T_2 according to Definition 1.1(ii), with $T_i \in B(n_i)$, $i = 1, 2$. Then

$$\varphi_n(T) = \begin{cases} 0 & n = 0 \\ n_1 - 1 & \\ \sum_{v=0}^{n_1-1} B(v) B(n-v-1) + \varphi_{n_1}(T_1) B(n_2) & \\ & + \varphi_{n_2}(T_2) \quad n \geq 1 \end{cases} \quad (2.7)$$

2.8 Lemma: φ_n is the ranking function on $B(n)$ with respect to the pre-order ordering.

Proof: By the remarks in §1, it suffices to show that

$$\varphi_n(T) = |\{T' \in B(n) : T' < T\}|,$$

where $<$ denotes the pre-order ordering on $B(n)$. The proof is by induction on n ; the case $n = 0$ is obvious.

Let $n \geq 1$. Let $T' < T$, with T_1', T_2' as in Definition 1.1 (ii).

It follows from Definition 2.4 (ii) that $T_1' \leq T_1$.

Case I: $T_1' \in B(v)$, $v < n$. There are $n_1 - 1$ choices for v ; for each v , there are $B(v)$ choices for T_1' ; and for each choice of T_1' , there are $B(n-v-1)$ choices for T_2' . Thus there are a total of

$$\sum_{v=0}^{n_1-1} B(v) B(n-v-1)$$

occurrences of Case I.

Case II: $T_1' \in B(n_1)$, $T_1' < T_1$. By the induction hypothesis there are $\varphi_{n_1}(T_1)$ choices for T_1' ; and there are $B(n_2)$ choices T_2' . Thus there are

a total of

$$\varphi_{n_1}(T_1) B(n_2)$$

occurrences of Case II.

Case III: $T_1' = T_1$. Then $T_2' \in B(n_2)$ and $T_2' < T_2$. By the

induction hypothesis there are $\varphi_{n_2}(T_2)$

occurrences of Case III.

Cases I, II, and III are mutually exclusive and exhaust the possibilities for $T' < T$ in $B(n)$

q.e.d.

Equations 2.7 provide the basis for an algorithm which computes $\varphi_n(T)$ for $T \in B(n)$. Assume that the vertices of T are labeled with the integers 1, ..., n . The algorithm uses five arrays: *lchild* [1: n], *rchild* [1: n], *b* [0: n], *d* [0: n], *phi* [0: n]. For $1 \leq v \leq n$, *lchild* [v] is the label of the left child of v if that child exists, and 0 otherwise; *rchild* [v] is defined similarly. *b* [j] = $B(j)$, $0 \leq j \leq n$. *d* [v] is the number of descendants of v , i.e. $d[v] = |T_v|$ where T_v is the subtree of T rooted at v , $1 \leq v \leq n$.

2.9 Algorithm: Input: a non-empty binary tree T represented by the arrays *lchild* [1: n], *rchild* [1: n]. Output: $\varphi_n(T)$ as defined by Equations 2.7.

begin

procedure *rank* (v);

begin

if *lchild* [v] $\neq$ 0 then *rank* (*lchild* [v]);

if *rchild* [v] $\neq$ 0 then *rank* (*rchild* [v]);

d [v] := *d* [*lchild* [v]] + *d* [*rchild* [v]] + 1;

t := 0;

```

for i: = 0 until d [lchild [v] - 1] do
    t: = t + b [i] * b [d[v]-i-1];
    phi [v]: = t + phi [lchild [v]] * b [rchild [v]] + phi [rchild [v]];
end rank (v);

d [0]: = phi [0]: = 0;
compute b [0:n];
rank (r);

end.

```

Note that the method by which the contents of $b [0:n]$ are computed is left unspecified; this will be dealt with shortly.

It is evident that Algorithm 2.9 is a pre-order search of the binary tree T with some additional computations; thus arises the name of the total order under discussion. The computations themselves are a straight-forward implementation of Equations 2.7.

The complexity of this algorithm is determined in the following theorem.

2.10 Theorem: Algorithm 2.9 is of $O(n^2)$ time complexity and $O(n)$ space complexity.

Proof: The contents of $b[0:n]$ may be computed using the recursion

$$B(0) = 1; B(n) = 2(2n+1) B(n)/(n+2), n \geq 1.$$

The time required is $O(n)$.

As noted above, the algorithm is a pre-order search of the binary tree T

with some additional computations; these computations will determine the time complexity of the algorithm. Let $t(n)$ be this complexity as a function of n .

From Algorithm 2.9, $t(n)$ satisfies the recursion

$$t(n) = t(n_1) + t(n_2) + O(n_1).$$

Assume for the moment that $t(n) = O(n^\alpha)$, $\alpha \geq 1$; then $t(n)$ is convex. See [R], p. 60 for a definition of convex function. It follows that

$$t(n_1) \leq \frac{n_1}{n-1} t(n-1) + \frac{n_2}{n-1} t(0);$$

and

$$t(n_2) \leq \frac{n_2}{n-1} t(n-1) + \frac{n_1}{n-1} t(0);$$

hence

$$t(n) \leq t(n-1) + O(n-1)$$

and thus

$$t(n) \leq O(n^2)$$

On the other hand, if T is the binary tree with n vertices satisfying

$$rchild[v] = 0, \quad 1 \leq v \leq n,$$

Algorithm 2.9 requires $O(n^2)$ time to compute $\varphi_n(T)$. Thus the time bound is established.

The space bound is obvious.

q.e.d.

An example will illustrate the action of Algorithm 2.9

2.13 Example: Let T be the five-vertex binary tree shown in Figure 2.13 (i). Algorithm 2.9 calls $rank(r)$; $rank(r)$ calls $rank(s)$

recursively, and $rank(s)$ calls $rank(t)$ recursively. Since $lchild[t] = rchild[t] = 0$, no recursive calls are made, and the computations in procedure $rank$ are executed. The results of these computations are

$$d[t] = 1,$$

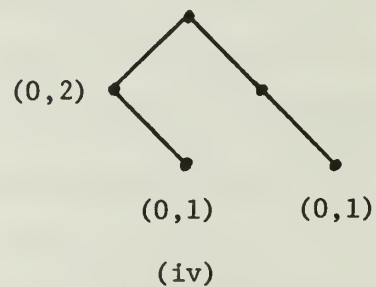
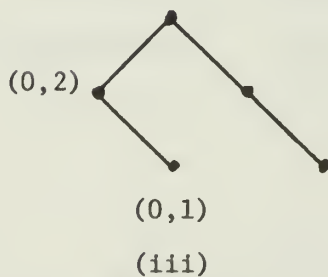
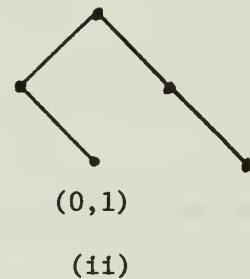
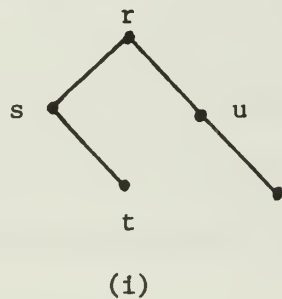
$$phi[t] = 0.$$

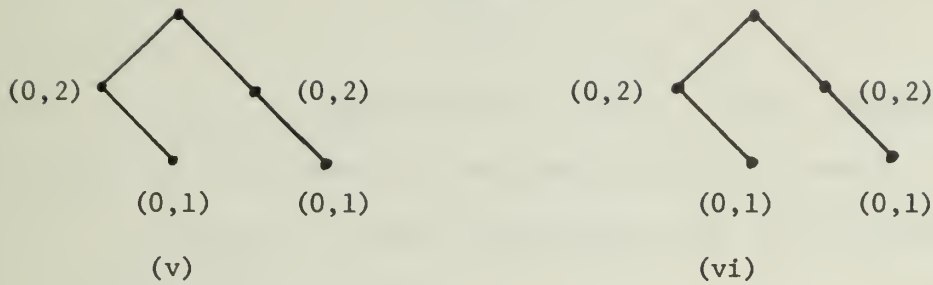
These results are shown in Figure 2.13 (ii) as the ordered pair beneath vertex t . The completion of $rank(t)$ allows $rank(s)$ to be completed; the following computation are performed:

$$d[s] = 0 + 1 + 1 + = 2;$$

$$phi[s] = 0;$$

the results are recorded in Figure 2.13 (iii).





2.13 Figure: Successive stages in the computation of the rank of T.

The completion of *rank* (s) allows *rank* (r) to call *rank* (u), which in turn calls *rank* (v). The resulting computations up to the conclusion of *rank* (u) are similar to those just described, and the results are shown in Figure 2.13 (iv) and Figure 2.13 (v). The completion of *rank* (u) allows the computations in *rank* (r) to take place; these computations are:

$$\begin{aligned}
 d[r] &= 2 + 2 + 1 = 5; \\
 \text{phi}[r] &= b[0]*b[4] + b[1]*b[3] \\
 &\quad + \text{phi}[s]*b[2] + \text{phi}[u] \\
 &= 14 + 5 + 0 + 0 = 19.
 \end{aligned}$$

Hence T is the rank 19 binary tree with 5 vertices.

Now the converse problem will be considered; that is, given $n \geq 0$ and m with $0 \leq m \leq B(n) - 1$, construct the rank m binary tree with n vertices. The next algorithm solves this problem.

Before the presentation of the algorithm, some comments are in order. The algorithm below works with the same four arrays as Algorithm 2.9, except that the zero entries of *d* and *phi* are not used in this algorithm. It is readily shown by a simple induction that the binary

tree produced by this algorithm, represented by the arrays *lchild* [1:n] and *rchild* [1:n], is labeled in preorder.

2.14 Algorithm: Input: $n \geq 0$, m satisfying $0 \leq m \leq B(n) - 1$. Output: The rank m binary tree with n vertices, represented by the arrays *lchild* [1:n], *rchild* [1:n], with the vertices labeled in preorder.

begin

procedure *unrank* (v, c);

begin

if $v \geq 1$ then

if $c > 1$ then

begin

$t := 0$

for $j := 0$ step 1 while $\phi[v] - t \geq b[j] * b[c-j-1]$ do

$t := t + b[j] * b[c-j-1];$

if $j = 0$ then *lchild* [v]: = 0

else *lchild* [v]: = $i := i + 1$;

if *lchild* [v] ≥ 1 then

begin

$d[lchild[v]] := j;$

$\phi[lchild[v]] := (\phi[v] - t) \div b[j];$

end;

unrank (*lchild*[v], $d[lchild[v]]$);

```

if  $j = c - 1$  then  $rchild[v]: = 0$ 

else  $rchild[v]: = i: = i + 1;$ 

if  $rchild[v] \geq 1$  then

  begin

     $d[rchild[v]]: = c-j-1;$ 

     $phi[rchild[v]]: = (phi[v]-t) \bmod b[j];$ 

  end;

   $unrank(rchild[v], d[rchild[v]]);$ 

end;

else  $lchild[v]: = rchild[v]: = 0;$ 

end  $unrank(v, c);$ 

 $i: = 1;$ 

 $phi[1]: = m;$ 

 $d[1]: = n;$ 

 $compute\ b[0:n];$ 

 $unrank(1, d[1]);$ 

end

```

The complexity of this algorithm is analyzed in the next result.

2.15 Theorem: Algorithm 2.14 is $O(n^2)$ time-bounded and $O(n)$ space-bounded.

Proof: Let $t(n)$ be the time bound. The procedure call

unrank (1,n) results in $O(n_1)$ computations plus two recursive calls of *unrank*. Thus, the function $t(n)$ satisfies the recursion

$$t(n) = t(n_1) + t(n_2) + O(n_1);$$

and thus

$$t(n) \leq O(n^2)$$

as in Theorem 2.10.

Substituting $m = B(n) - 1$ results in

$$t(n) \geq O(n^2),$$

and the time bound is established.

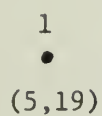
The space bound is obvious.

q.e.d.

An example will serve to illustrate the operation of the algorithm.

2.16 Example: Apply Algorithm 2.14 to the numbers $n = 5$, $m = 19$. At the outset, the situation is that depicted in Figure 2.17 (i); a root vertex labeled 1 with $d[1] = 5$, $\phi[1] = 19$. These latter values are given in parentheses. The first procedure call is *unrank* (1,5). The loop on j computes $d[lchild[1]]$; in this case the value is 2, which is non-zero, so *lchild* [1] is assigned the label 2, and $\phi[2]$ is assigned the value 0 (Figure 2.17 (ii)). Since $d[2] > 1$, *unrank* (2,2) is called recursively. The loop on j computes $d[lchild[2]] = 0$; hence $d[rchild[2]] = 1$ and this vertex is labeled 3 (Figure 2.17 (iii)). The recursive call *unrank* (3,1) has no effect except setting *rchild* [3] and *lchild* [3] to zero. The recursive calls are completed, so the next computation is to assign *rchild* [1] the value 4, $d[4]$ the value 2, and $\phi[4]$ the value

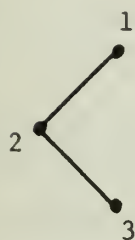
zero (Figure 2.17 (iv)). *unrank* (4,2) is called, and computations proceed as they did for the call *unrank* (2,2) (Figure 2.17 (v)-(vi)).



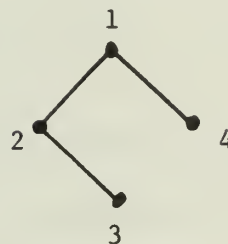
(i)



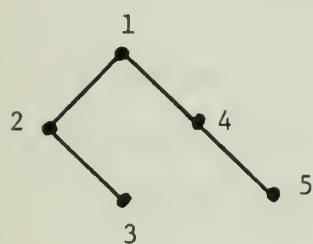
(ii)



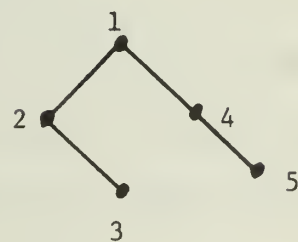
(iii)



(iv)



(v)



(vi)

§III. Preorder Ranking of k-ary Trees: The development in this section is exactly parallel with that of §II; hence many of the details will be omitted.

3.1 Definition [$\tilde{S}(k,n)$, $S(k,n)$]: Let n be a non-negative integer, $k \geq 2$.

Then

$$\tilde{S}(k,n) = \{k\text{-ary trees with } n \text{ vertices}\};$$

and

$$S(k,n) = |\tilde{S}(k,n)|.$$

$$S(k,0) = S(k,1) = 1. \quad \text{For } n \geq 2, \text{ we have}$$

$$S(k,n) = \sum_{v_1+v_2+\dots+v_k = n-1} S(k,v_1) S(k,v_2) \dots S(k,v_k); \quad (3.2)$$

this follows from Definitions 1.1 and 1.4. A generating function may be constructed from Equation 3.2; and application of other results to this generating function yields

$$S(k,n) = \frac{1}{(k-1)n+1} \binom{kn}{n}; \quad (3.3)$$

see p. 584 of [K]. An independent derivation of Equation (3.3) will be given in §V below. It should be noted for later reference that the summation on the right of Equation (3.2) has $\binom{n+k-2}{k-1}$ terms.

3.4 Definition [Pre-Order Ordering]: Let T_1 and T_2 be k -ary trees with n_1 and n_2 vertices respectively; let $T_{11}, \dots, T_{1k}, T_{21}, \dots, T_{2k}$ be defined according to Definition 1.1 (ii). Then $T_1 < T_2$ iff

- (i) $n_1 < n_2$; or
- (ii) for some j , $1 \leq j \leq k-1$, $T_{11} = T_{21}, T_{12} = T_{22}, \dots, T_{1,j-1} = T_{2,j-1}$, and $T_{1j} < T_{2j}$.

3.5 Lemma: Definition 3.4 defines a total ordering on the set of all k -ary trees.

Proof: Analogous to the proof of Lemma 2.5.

q.e.d.

As in §II, Definition 3.4 may be used to define a ranking function on $S(k, n)$.

3.6 Definition [Pre-Order Ranking]: Let $T \in S(k, n)$. If $n \geq 1$, define $T_1, T_2, \dots, T_k$ according to Definition 1.1 (ii), with $T_j \in S(k, n_j)$, $1 \leq j \leq k$.

Then

$$\varphi_{k, n}(T) = \begin{cases} 0 & n = 0 \\ \sum_{\substack{v_1 + v_2 + \dots + v_k = n - 1 \\ v_1 < n_1}} S(k, v_1) \dots S(k, v_1) \dots S(k, v_k) \\ + \varphi_{k, n_1}(T_1) \sum_{\substack{v_2 + v_3 + \dots + v_k = n - n_1 - 1 \\ v_2 < n_2}} S(k, v_2) \dots S(k, v_k) \\ + \dots \\ + \varphi_{k, n_{i-1}}(T_{i-1}) \sum_{\substack{v_i + v_{i+1} + \dots + v_k = n - n_1 - \dots - n_{i-1} - 1 \\ v_i < n_i}} S(k, v_i) \dots S(k, v_k) \\ + \dots \\ + \varphi_{k, n_k}(T_k) & n \geq 1 \end{cases} \quad (3.7)$$

3.8 Lemma: $\varphi_{k, n}$ is the ranking function on $S(k, n)$ with respect to the pre-order ordering.

Proof: Straight-forward generalization of the proof of Lemma 3.8.

q.e.d.

Equations 3.7 provide the basis for ranking and unranking algorithms analogous to Algorithms 2.9 and 2.14. These algorithms will not be presented explicitly since they are straight-forward generalizations of the above mentioned algorithms, and also because they are rather tedious to state in detail. Nevertheless, since these algorithms are so similar to Algorithm 2.9 and 2.14, their complexity may be determined as in Theorems 2.10 and 2.15.

3.9 Theorem: The ranking and unranking algorithms based on Equations 3.7 have time complexity $O(n^k)$ and space complexity $O(kn)$.

Proof: The computation of $S(k,v)$ requires $O(v)$ time; hence the computation of $S(k,0), \dots, S(k,n)$ requires a total of $O(n^2)$ time.

Let $t(n)$ be the time bound; suppose that $t(n) = O(n^\alpha)$, $\alpha \geq 1$.

In Equations 3.7, denote by Σ_0 the first summation, and by Σ_i the coefficient of $\varphi_k, n_i (T_i)$. Then Σ_0 has at most $O\left(\binom{n+k-2}{k-1}\right) = O(n^{k-1})$ terms, and Σ_i has at most $O\left(\binom{n-n_1-1 \dots - n_i-2}{k-i-1}\right) \leq O(n^{k-i-1})$ terms.

Thus

$$t(n) \leq \sum_{i=1}^k (t(n_i) + O(n^{k-1}))$$

Since n^α is convex for $\alpha \geq 1$,

$$t(n_i) \leq \frac{n_i}{n-1} t(n-1) + \frac{n-n_i-1}{n-1} t(0);$$

hence

$$t(n) \leq t(n-1) + \sum_{i=1}^k O(n^{k-1})$$

and thus

$$t(n) \leq O(n^k).$$

On the other hand, if $T \in \tilde{S}(k, n)$ has the property that for each $v \in V(T)$, $T_{vk} = \emptyset$ for $k \geq 2$, it is easy to see that the ranking algorithm requires $O(n^k)$ time to compute the rank of this tree. Hence

$$t(n) \geq O(n^k)$$

and the time bound is established.

The space bound for the ranking algorithm is clear.

The analysis of the unranking algorithm is similar.

q.e.d.

Thus the complexity of the ranking and unranking algorithms are exponential in k . In the next two sections, alternative ranking schemes will be developed which avoid this difficulty.

§IV. Lexicographic Ranking of Binary Trees: The discussion starts with a definition.

4.1 Definition [Permutation]: Let n be a positive integer. A permutation σ of $1, 2, \dots, n$ is a finite sequence $s_1 s_2 \dots s_n$ satisfying

$$\{s_1, \dots, s_n\} = \{1, \dots, n\}.$$

The set of all permutations of $1, 2, \dots, n$ is denoted by S_n .

There is much more structure which can be associated with S_n , but it will not be necessary for the present development.

4.2 Definition [$\sigma(j_1, \dots, j_m)$]: Let n and m be positive integers with $n \geq m$. Let $\{j_1, \dots, j_m\} \subseteq \{1, \dots, n\}$; let $\sigma \in S_n$. $\sigma(j_1, \dots, j_m)$ is the finite sequence obtained by deleting $j_1, \dots, j_m$ from σ .

4.3 Definition [$P(n)$]: n be a positive integer. $P(n) \subseteq S_n$ is defined recursively as follows:

$$(i) \quad P(1) = S_1;$$

$$(ii) \quad \text{If } n > 1, \sigma \in P(n) \text{ iff } \sigma(n) \in P(n-1) \text{ and } s_j = n, s_k = n-1 \\ \text{implies } j \geq k-1.$$

The following theorem gives the relationship between $P(n)$ and $B(n)$.

4.4 Theorem: There is a 1-1 correspondence between $P(n)$ and $B(n)$, $n \geq 1$.

Proof: Let $T \in B(n)$; generate a permutation $\sigma \in S_n$ according to the following scheme: label the vertices of T in pre-order, and read off the labels in in-order. It must be shown that $\sigma \in P(n)$. The proof is by induction on n .

For $n = 1, 2$, the result is immediate. Let $n > 2$, $T_1 \in \underline{B}(n_1)$, $T_2 \in \underline{B}(n_2)$ as in Definition 1.1, and assume that the result is true for $1, 2, \dots, n-1$.

Let $\sigma \in \underline{P}(n_1)$ be the permutation generated from T_1 , and $\tau \in \underline{P}(n_2)$ be the permutation generated from T_2 ; if $n_1 = 0$ or $n_2 = 0$, the corresponding permutation is defined to be the empty sequence. Let $\sigma = s_1 s_2 \dots s_{n_1}$,

$\tau = t_1 t_2 \dots t_{n_2}$. Let $s_j' = s_j + 1$; let $t_j' = t_j + n_1 + 1$. Then it is

clear that the permutation ρ generated from T is given by

$$\rho = s_1' s_2' \dots s_{n_1}' 1 t_1' t_2' \dots t_{n_2}'. \quad (4.5)$$

It follows that $s_i' = n$, $t_j' = n-1$ is impossible, and since $\sigma \in \underline{P}(n_1)$,

$\tau \in \underline{P}(n_2)$, the second condition of Definition 4.3 (ii) must be satisfied.

Let $\rho = r_1 r_2 \dots r_n$, with $r_j = n$. Let $v \in V(T)$ be the vertex labeled n .

If $T' \in \underline{B}(n-1)$ is the binary tree obtained by deleting v from T , the

permutation generated from T' is $\rho(n)$, and by the induction hypothesis,

$\rho(n) \in \underline{P}(n-1)$; hence the first condition of Definition 4.3 (ii) is satisfied.

Thus the proof that $\sigma \in \underline{P}(n)$ is complete.

So far, the existence of a 1-1 map from $\underline{B}(n)$ into $\underline{P}(n)$ has been proven; it remains to show that this map is onto. This will be accomplished by constructing an inverse to the map just defined.

Let $a_1 a_2 \dots a_n$ be a finite sequence of distinct positive integers.

An unique binary tree $T(a_1 a_2 \dots a_n) \in \underline{B}(n)$ may be constructed as follows:

(I) If $n = 1$, $T(a_1)$ is the binary tree with one vertex;

(II) if $n > 1$, let $a_j = \min\{a_1, \dots, a_n\}$. Then $T(a_1, a_2 \dots a_m)$ is the binary tree formed by joining $T(a_1 a_2 \dots a_{j-1})$ and $T(a_{j+1} a_{j+2} \dots a_n)$ to a root r as in Definition 1.1 (ii); if either sequence is empty, so is the corresponding tree.

The claim is that $\rho \mapsto T(\rho)$ is the inverse of the map previously constructed; the proof is by induction on n .

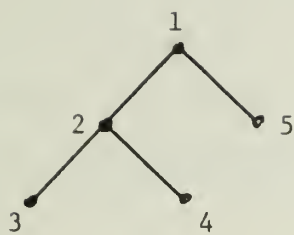
For $n = 1, 2$, the result is obvious. Suppose that $n > 2$, and that the result holds for $1, 2, \dots, n-1$. Let $\sigma \in \tilde{P}(n_1)$; $\tau \in \tilde{P}(n_2)$, s_i, t_j be as before. Referring to Equation 4.5, it is easy to see that $T(s_1' \dots s_{n_1}') = T(\sigma)$ and $T(t_1' \dots t_{n_2}') = T(\tau)$; thus $T(\rho)$ is the binary tree formed by joining $T(\sigma)$ and $T(\tau)$ to a new root; but by the induction hypothesis, $T(\sigma) = T_1$, $T(\tau) = T_2$. Hence $T(\rho) = T$.

q.e.d.

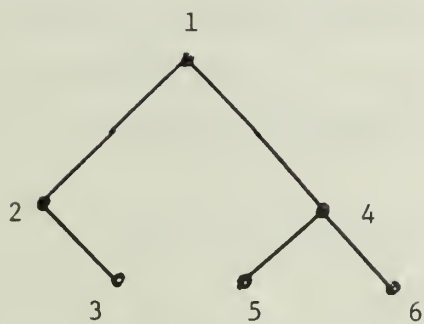
This same result, using a different (although equivalent) definition of $\tilde{P}(n)$ is derived in Exercises 2.2.1-5 and 2.3.1-6 of [K]. This derivation goes by way of stacks, while the above theorem obtains the result directly.

4.6 Example: For $n = 1, 2$, $\tilde{P}(n) = \tilde{S}_n$. For $n = 3$, $\tilde{P}(n) = \tilde{S}_n - \{312\}$. $4321 \in \tilde{P}(4)$ while $4312 \notin \tilde{P}(4)$. The permutation generated from the tree in Figure 4.7(i) is 32415; the permutation generated from the tree in Figure 4.7(ii) is 231546.

4.8 Definition [Lexicographic Ordering on $\tilde{B}(n)$]: Let n be a positive integer; let $T_1, T_2 \in \tilde{B}(n)$; T_1 is lexicographically less than T_2 iff the permutation $\sigma_1 \in \tilde{P}(n)$ corresponding to T_1 is lexicographically less than $\sigma_2 \in \tilde{P}(n)$ corresponding to T_2 .



(i)



4.7 Figure

The remainder of this section will be concerned with ranking and unranking algorithms on $\tilde{P}(n)$ with respect to the lexicographic ordering.

4.9 Definition $[\tilde{P}(n,m), \tilde{P}(n,m)]$: Let n be a positive, m a non-negative integer; define

$$\tilde{P}(n,m) = \{\sigma \in \tilde{P}(n) : s_m = n\};$$

and

$$P(n,m) = |\tilde{P}(n,m)|.$$

Clearly $P(n,m) = 0$ unless $1 \leq m \leq n$.

From this point on, the notion of concatenation of lists or strings will be important; the symbol $||$ will be used to denote the operation of concatenation of lists or strings.

4.10 Definition [Direct Insertion Order on $\tilde{S}_n, \tilde{P}(n)$]: Let n be a positive

integer. The direct insertion order is defined on $\tilde{S}_n$ as follows:

(i) $\tilde{S}_1$ is in direct insertion order;

(ii) Let $L_{n-1} = (\sigma_1, \sigma_2, \dots, \sigma_{(n-1)!})$ be a listing of $\tilde{S}_{n-1}$

in direct insertion order. For $1 \leq i \leq (n-1)!$, create a list $L_{n,i}$ by inserting the character n into each possible position in σ_i , starting at the right and working leftwards. The listing of $\tilde{S}_n$ in direct insertion order is

$$L_{n,1} || L_{n,2} || \dots || L_{n,(n-1)!}.$$

The definition of direct insertion order on $\tilde{P}(n)$ is similar except that in part (ii), if $\sigma_i \in \tilde{P}(n-1,m)$, the insertion process halts with the insertion

of n to the immediate left of s_m .

Figure 4.11 depicts the generation of $S_{\sim n}$ and $P(n)$ by direct insertion, for $n = 1, 2, 3, 4$.

Note that while lexicographic order and direct insertion order differ on $S_{\sim n}$ for $n = 3, 4$, they coincide on $P(n)$ for $n = 1, 2, 3, 4$. This is not accidental.

4.12 Lemma: The lexicographic and direct insertion orders on $P(n)$ coincide.

Proof: Induction on n ; from Figure 4.11 the result is true for $n = 1, 2, 3, 4$.

Suppose the result is true for $P(n-1)$; let $L_{n,i}$ be the list generated from σ_i , $1 \leq i \leq B(n-1)$. Clearly each $L_{n,i}$ is in lexicographic order; so it need only be shown that the last element of $L_{n,i}$ is lexicographically less than the first element of $L_{n,i+1}$, $1 \leq i \leq B(n-1) - 1$.

Let $\sigma_i = s_{i,1} s_{i,2} \dots s_{i,n-1}$, with $\sigma_i \in P(n-1, m)$; and let $\sigma_{i+1} = s_{i+1,1} s_{i+1,2} \dots s_{i+1,n-1}$. Then the last element of $L_{n,i}$ is $\rho = s_{i,1} \dots s_{i,m-1}^{(n)} s_{i,m} \dots s_{i,n-1}$; and the first element of $L_{n,i+1}$ is $\tau = s_{i+1,1} \dots s_{i+1,n-1}^{(n)}$.

First note that since it is assumed that $i \leq B(n-1) - 1$, it must be that $2 \leq m \leq n - 1$, for the only element of $P(n-1, 1)$ is $(n-1) (n-2) \dots (2) (1) = \sigma_{B(n-1)}$.

Suppose that $\tau < \rho$ lexicographically in $P(n)$. Since $\sigma_i < \sigma_{i+1}$ lexicographically in $P(n-1)$, it follows that $s_{i,j} = s_{i+1,j}$ for $1 \leq j \leq m-1$,

	n = 1	2	3	4
S_n	1	12 21	123 132 312 213 231 321	1234 1243 1423 4123 1324 1342 1432 4132 3124 3142 3412 4312 etc.
$P(n)$	1	12 21	123 132 213 231 321	1234 1243 1324 1342 1432 2134 2143 2314 2341 2431 3214 3241 3421 4321

4.11 Figure: Generation of S_n and $P(n)$ by direct insertion.

and $s_{i,m} \leq s_{i+1,m} < n$. But since ρ is the lexicographically largest element of $L_{n,i}$, it follows that $s_{i,m} = n - 1$; hence $s_{i+1,m} = n - 1$. But σ_i and σ_{i+1} are successive elements in the list of S_{n-1} in direct insertion order; hence $s_{i,m} = s_{i+1,m} = n - 1$ is impossible.

q.e.d.

Before proceeding with the development of the ranking and unranking algorithms, it is worthwhile to evaluate $P(n,m)$ for $1 \leq m \leq n$. Considering the construction of $P(n)$ from $P(n-1)$ by direct insertion, it is easy to see that for $m \geq 1$, $n \geq 2$,

$$P(n,m) = \sum_{\mu=1}^m P(n-1,\mu). \quad (4.13)$$

As was remarked in the proof of Lemma 4.12,

$$P(n,1) = 1, \quad n \geq 1. \quad (4.14)$$

Equations (4.13) and (4.14) may be used to obtain the following recursion with boundary conditions

$$P(n,m) = P(n,m-1) + P(n-1,m), \quad n \geq 2, \quad m \geq 1; \quad (4.15)$$

$$P(n,1) = 1, \quad n \geq 1;$$

$$P(n,m) = 0 \quad n < m.$$

4.16 Lemma: For $2 \leq m \leq n$,

$$P(n,m) = \frac{(n-m+1)}{(m-1)!} \prod_{\mu=1}^{m-2} (n+\mu), \quad (4.17)$$

where the product is defined to be 1 for $m = 2$.

Proof: In the case $m = 2$, the formula reduces to $P(n,2) = n - 1$, which can be verified from Equations (4.15). The proof is by induction on

m and n .

Suppose that $3 \leq m \leq n$, that $P(n, m-1) = \frac{(n-m+2)}{(m-2)!} \prod_{n=1}^{m-3} (n+\mu)$, and that

$$P(n-1, m') = \frac{(n-m'+2)}{(m-1)!} \prod_{\mu=1}^{m-2} (n+\mu),$$

$$2 \leq m' \leq m-1.$$

Then

$$\begin{aligned} P(n, m) &= P(n, m-1) + P(n-1, m) \\ &= \frac{(n-m+2)}{(m-2)!} \prod_{\mu=1}^{m-3} (n+\mu) + \frac{(n-m)}{(m-1)!} \prod_{\mu=1}^{m-2} (n+\mu-1) \\ &= \frac{(n-m+2)}{(m-2)!} \prod_{\mu=1}^{m-3} (n+\mu) + \frac{(n-m)}{(m-1)!} \prod_{\mu=0}^{m-3} (n+\mu) \\ &= \left(\frac{n-m+2}{(m-2)!} + n \cdot \frac{n-m}{(m-1)!} \right) \prod_{\mu=1}^{m-3} (n+\mu) \\ &= \left(\frac{(m-1)(n-m+2) + n(n-m)}{(m-1)!} \right) \prod_{\mu=1}^{m-3} (n+\mu) \\ &= \left(\frac{(n+m-2)(n-m+1)}{(m-1)!} \right) \prod_{\mu=1}^{m-3} (n+\mu) \\ &= \frac{(n-m+1)}{(m-1)!} \prod_{\mu=1}^{m-2} (n+\mu). \end{aligned}$$

q.e.d.

This derivation allows a proof of the next corollary without reference to generating functions, Taylor's series, etc.

4.18: Corollary: $B(n) = P(n) = \frac{1}{n+1} \binom{2n}{n}.$

Proof: It is easy to see that $P(n) = P(n+1, n+1)$, and substituting in Equation 4.17 yields the desired result.

q.e.d.

For $\sigma \in \tilde{P}(n)$, denote by $\text{lexrank}(\sigma)$ the rank of σ in the lexicographic ordering of $\tilde{P}(n)$. It will be seen in the next lemma that the $P(i, j)$ are intimately involved in the computation of $\text{lexrank}(\sigma)$.

4.19 Lemma: Let $\sigma = s_1 s_2 \dots s_n \in \tilde{P}(n)$, let $0 \leq j < i < n$, and suppose that σ satisfies

$$s_\ell = \ell, \quad n - j + 1 \leq \ell \leq n;$$

$$s_{n-i} = n - j.$$

Let $\tau = s_1 s_2 \dots s_{n-i-1} s_{n-i+1} \dots s_{n-j} (n-j) (n-j+1) \dots (n)$.

Then $\text{lexrank}(\sigma) - \text{lexrank}(\tau) = P(i+1, j+2)$.

Proof: By induction on i, j . If $j = 0$, $s_n \neq n$, and clearly $\text{lexrank}(\sigma) - \text{lexrank}(\tau) = i = P(i+1, 2)$.

Suppose that $j > 0$ and $i = j + 1$. Then

$$\sigma = s_1 \dots s_{n-j-2} (n-j) s_{n-j} (n-j+1) \dots (n);$$

$$\tau = s_1 \dots s_{n-j-2} s_{n-j} (n-j) (n-j+1) \dots (n).$$

Let

$$\sigma' = s_1 \dots s_{n-j-2} s_{n-j} (n) (n-1) \dots (n-j+1) (n-j).$$

Considering the insertion process by which σ' and σ are constructed from $s_1 \dots s_{n-j-2} s_{n-j} \in \tilde{P}(n-j-1)$, it is seen that

$$\text{lexrank}(\sigma) = \text{lexrank}(\sigma') + 1$$

Thus

$$\begin{aligned} \text{lexrank}(\sigma) - \text{lexrank}(\tau) &= 1 + \text{lexrank}(\sigma') \\ &\quad - \text{lexrank}(\tau); \end{aligned}$$

but $\{\rho: \tau < \rho < \sigma \text{ lexicographically}\}$ is order isomorphic to $P(j+1)$. Hence

$$\begin{aligned} \text{lexrank}(\sigma) - \text{lexrank}(\tau) &= 1 + P(j+1) - 1 \\ &= P(j+1) \\ &= P(j+2, j+2). \end{aligned}$$

Now let $j > 0$, $i > j + 1$, and suppose inductively that the result is true for $i - 1$ and j ; and that the result is true for $j - 1$ and all i' satisfying $j - 1 < i' < n$

Let

$$\sigma = s_1 s_2 \cdots s_{n-j-1} (n-j) s_{n-i+1} \cdots s_{n-j} (n-j+1) \cdots (n);$$

then

$$\tau = s_1 s_2 \cdots s_{n-i-1} s_{n-i+1} \cdots s_{n-j} (n-j) (n-j+1) \cdots (n).$$

Let

$$\sigma' = s_1 s_2 \cdots s_{n-i-1} s_{n-i+1} (n) (n-1) \cdots (n-j) s_{n-i+2} \cdots s_{n-j};$$

$$\sigma'' = s_1 s_2 \cdots s_{n-i-1} s_{n-i+1} (n-j) s_{n-i+2} \cdots s_{n-j} (n-j+1) \cdots (n).$$

as before,

$$\text{lexrank}(\sigma) - \text{lexrank}(\sigma') = 1;$$

thus

$$\begin{aligned} \text{lexrank}(\sigma) - \text{lexrank}(\tau) &= 1 + \text{lexrank}(\sigma') - \text{lexrank}(\tau) \\ &= 1 + \text{lexrank}(\sigma') - \text{lexrank}(\sigma'') \\ &\quad + \text{lexrank}(\sigma'') - \text{lexrank}(\tau) \end{aligned}$$

But

$$\text{lexrank}(\sigma'') - \text{lexrank}(\tau) = P(i, j+2)$$

by the induction hypothesis on i ; and

$$\text{lexrank}(\sigma') - \text{lexrank}(\sigma'') = \sum_{v=0}^{j-1} P(i, v+2)$$

by the induction hypothesis on j .

Hence

$$\begin{aligned}
 \text{lexrank}(\sigma) - \text{lexrank}(\tau) &= 1 + \sum_{v=0}^{j-1} P(i, v+2) + P(i, j+2) \\
 &= \sum_{v=-1}^j P(i, v+2) \\
 &= P(i+1, j+2)
 \end{aligned}$$

by Equation 4.13.

q.e.d.

The result of Lemma 4.19 is exploited by the following algorithm, which computes the lexicographic rank of $\sigma \in P(n)$. In this algorithm σ is represented by a doubly linked list, although the representation and linked list manipulations are suppressed in the description presented here; the array $p[1:n, 1:n]$ is used to store the values of $P(i, j)$: $p[i, j] = P(i, j)$. Discussion of the method of computing the $p[i, j]$ is deferred until after the description of the algorithm.

4.20 Algorithm: Input: $\sigma \in P(n)$, represented by a doubly linked list.

Output: The lexicographic rank of σ in $P(n)$.

begin

procedure *lexrank* (σ, n, l);

begin

temp: = 0;

for j : = l until $n - 2$ do

if $s_{n-j} \lceil = n - j$ then

for i : = 1 until $n - 1$ do

```

if  $s_{n-i} = n - j$  then

begin

    temp: = p [i+1,j+2]

        + lexranks ( $\sigma(n-j, \dots, n) \parallel (n-j) \parallel \dots \parallel (n), n, j + 1$ );

    i: = n;

    j: = n - 1;

end;

    lexranks: = temp;

end lexranks ( $\sigma, n, l$ );

compute p[1:n, 1:n];

lexranks ( $\sigma, n, 0$ );

end.

```

4.21 Theorem: Algorithm 4.20 is $O(n^2)$ time-bounded; the algorithm can be implemented so that it is $O(n)$ space-bounded.

Proof: The procedure call $lexranks(\sigma, n, l)$ requires $O(n-1)$ steps to execute the loops on i and j in the worst case; so the worst-case time-complexity for $lexranks$ is given by the recursion

$$t(\text{lexranks}(\sigma, n, 0)) \leq O(n) + t(\text{lexranks}(\sigma(n) \parallel (n), n, 1))$$

since computation of $\sigma(n-j, n \dots n) \parallel (n-j) \parallel \dots \parallel \dots \parallel (n)$ is constant time by linked-list operations. But the time required by $lexranks(\sigma(n) \parallel (n), n, 1)$ is the same as the time required by $lexranks(\sigma(n), n-1, 0)$. Hence

$$t(\text{lexranks}(\sigma, n, 0)) = O(n^2).$$

Computation of $p[1:n, 1:n]$ by Equations 4.15 requires $O(n^2)$ time. It appears at first storing the $P(i, j)$ in the array $p[1:n, 1:n]$ (as opposed

to direct computation from Equation 4.17, which also leads to an $O(n^2)$ time-bound) requires $O(n^2)$ storage. However, any particular column of $p[1:n, 1:n]$ is accessed at most once during the execution of Algorithm 4.20; and those columns which are accessed in ascending order on j . Furthermore, if the j th column of $p[1:n, 1:n]$ is known, the $j + 1^{\text{st}}$ may be computed using Equations 4.17. Hence no more than two complete columns of $p[1:n, 1:n]$ need be kept in storage at any time; hence the space bound.

q.e.d.

The essential steps of Algorithm 4.20 are illustrated in the following example.

4.22 Example: Compute the lexicographic rank of $\sigma = 35421$ in $P(5)$.

According to Lemma 4.20,

$$\begin{aligned}
 \text{lexrank}(35421) &= \text{lexrank}(34215) + P(4,2) \\
 &= \text{lexrank}(32145) + P(4,3) + P(4,2) \\
 &= \text{lexrank}(21345) + P(5,4) + P(4,3) + P(4,2) \\
 &= \text{lexrank}(12345) + P(5,5) + P(5,4) + P(4,3) \\
 &\quad + P(4,2).
 \end{aligned}$$

But $\text{lexrank}(12345) = 0$; and evaluation of the appropriate $P(i,j)$ yields

$$\begin{aligned}
 \text{lexrank}(35421) &= 14 + 14 + 5 + 3 \\
 &= 36
 \end{aligned}$$

Lemma 4.20 also leads to an unranking algorithm.

4.23 Algorithm: Input: integers $n \geq 1$, $0 \leq r \leq B(n) - 1$. Output: the rank r element of $P(n)$, represented as a linked list.

begin

procedure *lexrecover* (r, n, j);

begin

```

if  $j \geq 0$  then
  begin
    find largest  $i$  such that  $P[i+1, j+2] \leq r$ ;
     $\sigma := s_1 \dots s_{n-i-1} (n-j) s_{n-i} \dots s_{n-j-1}$ ;
    lexrecover [ $r-p[i+1, j+2], n, j-1$ ];
  end;
end lexrecover ( $r, n, j$ );
compute  $p[1:n, 1:n]$ ;
 $\sigma := 1$ ;
lexrecover ( $r, n, n-2$ );
end.

```

In this algorithm, as in Algorithm 4.20, the linked-list operations have been suppressed.

4.24 Theorem: Algorithm 4.23 is $O(n^2)$ time-bounded and $O(n)$ space bounded.

Proof: In the procedure call *lexrecover* (r, n, j), the time required to find the appropriate $P(i, j)$ is at most $O(n-j)$; hence the time required for *lexrecover* (r, n, j) is given by

$$t(\text{lexrecover}(r, n, j)) \leq O(n-j) + t(\text{lexrecover}(r-P[i+1, j+2], n, j-1)).$$

It follows that the call *lexrecover* ($r, n, n-2$) requires at most $O(n^2)$ time. On the other hand, it can be seen that *lexrecover* ($B(n) - 1, n, n-2$) requires $O(n^2)$ time.

Computation of $P[1:n, 1:n]$ requires $O(n^2)$ time using Equations 4.15.

The $O(n)$ space requirement is demonstrated in a fashion similar

to the proof of Theorem 4.21; in this case, the columns of $P[1:n, 1:n]$ are accessed in decreasing order of j ; if the j^{th} column is known, the $j - 1^{\text{st}}$ can be computed using Equations 4.15.

q.e.d.

4.25 Example: Given $n = 5$, $r = 36$, compute the rank r element of $P(n)$ in lexicographic order. At the outset, set $\sigma = 1$. The largest i such that $P(i+1,5) \leq 36$ is 4: $P(5,5) = 14$. Hence 2 is inserted immediately to the left of $s_{5-4} = s_1$ in σ i.e. $\sigma = 21$; r is decremented to 22. The largest i such that $P(i+1,4) \leq 22$ is 4: $P(5,4) = 14$. Hence 3 is inserted into the first position in σ , i.e. $\sigma = 321$, and r is decremented to 8. At the next step, $j = 1$ and i is found to be 3; so σ becomes 3421 and r is decremented to 3; at the last step, $j = 0$, and i is found to be 3; so σ becomes 35421 and r is decremented to 0. The algorithm halts at this point.

In the next section, the ranking and unranking of k -ary trees will be treated in a similar fashion.

§V. Lexicographic Ranking of Full k-ary Trees.

Recall from Lemma 1.6 the correspondence between k-ary trees and full k-ary trees; from this correspondence, it follows that to rank k-ary trees on n vertices, it suffices to rank full k-ary trees on $kn + 1$ vertices. The general strategy will be to relate full k-ary trees on $kn + 1$ vertices to a subset of $\underline{B}(nk+1)$, which subset will be ranked in a manner similar to the lexicographic ranking of $\underline{B}(n)$ in §IV.

Using the standard definition of (ordered) tree and forest (see §2.3.2 of [K]), the following definition is made.

5.1 Definition [Operations on Ordered Forests]: Let $F = (T_1, \dots, T_n)$ $F' = (T_1', \dots, T_m')$ be ordered forests; then FF' is the ordered forest $(T_1, \dots, T_n, T_1', \dots, T_m')$; $\hat{F}$ is the ordered forest (T) , where T is the tree obtained by joining the root of each T_1 to a new root r . In particular, if F is the empty ordered forest, $\hat{F}$ is the ordered forest on one vertex.

Definition 5.1 allows a concise statement of the correspondence between ordered forests on n vertices and elements of $\underline{B}(n)$.

5.2 Definition [Correspondence Between Binary Trees and Ordered Forests]:

- (i) The empty binary tree corresponds to the empty ordered forest;
- (ii) If T_1 and T_2 are binary trees corresponding to the ordered forests F_1 and F_2 respectively, the binary tree T which has T_1 and T_2 as its left and right subtrees corresponds to $F_1 \hat{F}_2$.

The correspondence in §2.3.2 of [K] has the binary tree T corresponding to $\hat{F}_1 F_2$; Definition 5.2 (ii) is preferable for the present purposes. That the correspondence in Definition 5.2 is 1-1 follows from the fact that the correspondence in §2.3.2 of [K] is 1-1.

5.3 Definition [U_k]: Let $k \geq 2$; $U_k \in \tilde{B}(k)$ is the unique binary tree on k vertices such that no vertex of U_k has a right child.

5.4 Definition [$\tilde{F}(nk+1, k)$]: Let $n \geq 0$, $k \geq 2$.

(i) If $n = 0$, $\tilde{F}(1, k) = \tilde{B}(1)$;

(ii) If $n \geq 1$, let $n_1 + \dots + n_k = n - 1$, let $T_i \in \tilde{F}(n_i k + 1, k)$,

$1 \leq i \leq k$, and let T_i have root r_i ; form a binary tree $T \in \tilde{B}(nk+1)$ as follows:

(a) the root of T is r ; r has no left child, and the right child of r is r_1 ;

(b) for $1 \leq i \leq k$, the left child of r_i is r_{i+1} , if r_{i+1} exists; the right subtree of r_i is the right subtree of r_i in T_i .

Then $T \in \tilde{F}(nk+1, k)$.

(iii) $\tilde{F}(nk+1, k)$ has no other elements.

5.5 Example: Figure 5.6 depicts the sets $\tilde{F}(nk+1, k)$ for $k = 2, 3$, and $n = 0, 1, 2$.

5.7 Lemma: Let $n \geq 0$, $k \geq 2$. There is a 1-1 correspondence between $\tilde{F}(nk+1, k)$ and the set of full k -ary trees on $nk + 1$ vertices.

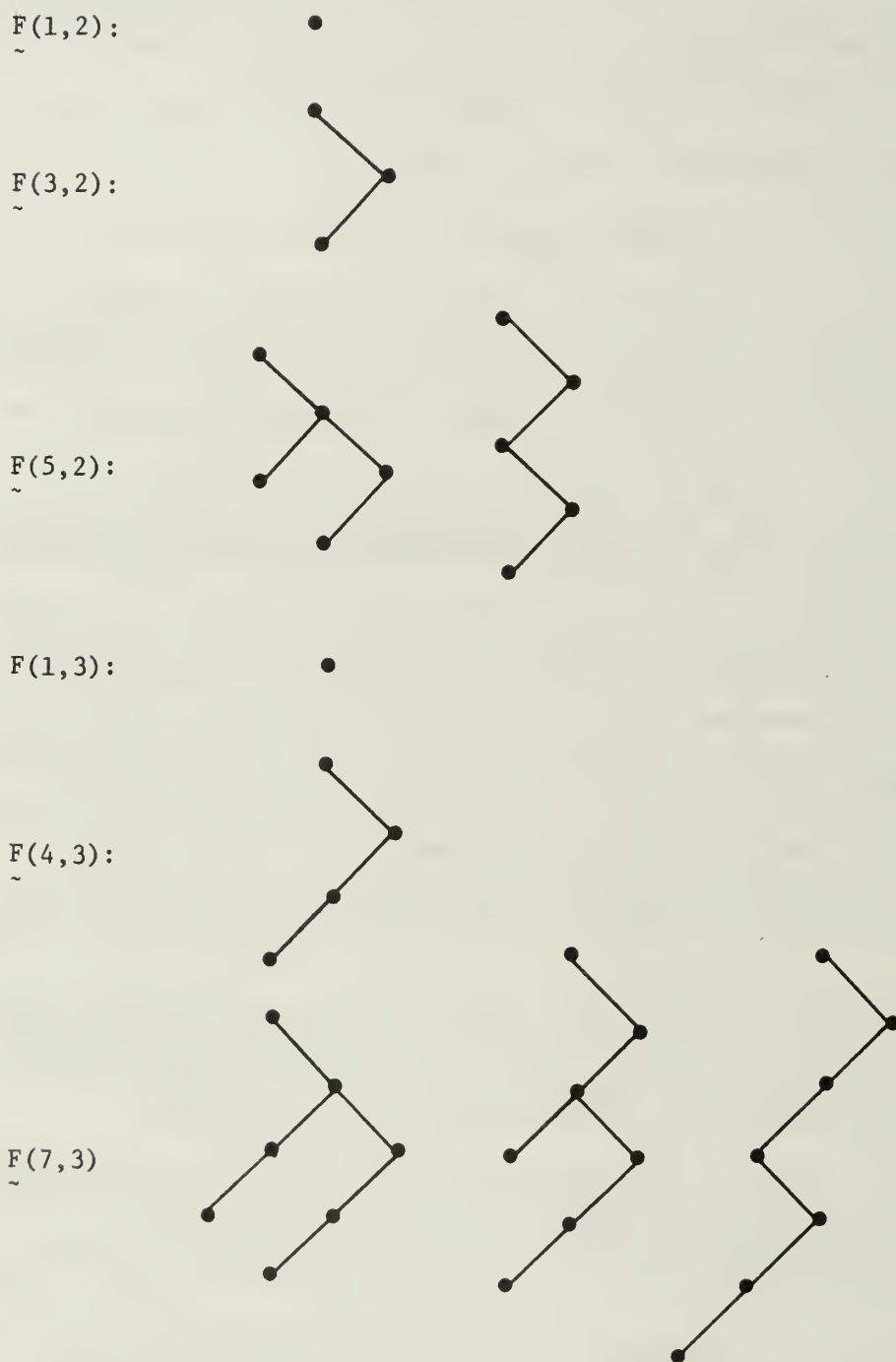
Proof: The 1-1 correspondence is the restriction to $\tilde{F}(nk+1, k)$ of the 1-1 correspondence in Definition 5.2. The proof is by induction on n .

The case $n = 0$ follows immediately from Definition 5.2(ii) applied to the binary tree on one vertex.

Suppose the result is true for $0, 1, \dots, n - 1$. Let

$T \in \tilde{F}(nk+1, k)$ be formed from $T_i \in \tilde{F}(n_i k + 1, k)$, $1 \leq i \leq k$ as in Definition

5.4. Let S_i be the full k -ary tree on $n_i k + 1$ vertices corresponding to



5.6 Figure: $F(nk+1,k)$ for $k = 2, 3, n = 0, 1, 2$.

T_1 . It follows from Definition 5.2 that the ordered forest corresponding to T is the tree S obtained by joining the root of each T_1 to a new root r ; hence S is a full k -ary tree on $nk + 1$ vertices. The argument is completely reversible.

q. e. d.

Thus, to rank the set of full k -ary trees on $nk + 1$ vertices, it suffices to rank $\tilde{F}(nk+1, k)$. This will be accomplished by considering the subset of $\tilde{P}(nk+1)$ generated by $\tilde{F}(nk+1, k)$ using the method of Theorem 4.4.

5.8 Definition $[\tilde{P}(nk+1; k)]: \tilde{P}(nk+1; k) \subseteq \tilde{S}_{nk+1}$ is the set of permutations generated from $\tilde{F}(nk+1, k)$ by the method of Theorem 4.4.

By what has already been said, $\tilde{P}(1; k) = \tilde{S}_1$; the next result characterizes $\tilde{P}(nk+1; k)$ for $n \geq 1$.

5.9 Lemma: Let $n \geq 1$, $k \geq 2$, $\sigma \in \tilde{S}_{nk+1}$. Then $\sigma \in \tilde{P}(nk+1; k)$ iff

$$(i) \quad \sigma((n-1)k + 2, (n-1)k + 3, \dots, nk, nk + 1) \in$$

$$\tilde{P}((n-1)k+1, k);$$

$$(ii) \quad (nk+1) (nk) (nk-1) \dots ((n-1)k + 2) \text{ is a substring of } \sigma;$$

$$(iii) \quad \text{if } s_m = nk + 1 \text{ and } s_{m'} = (n-1)k + 1, \text{ then } m > m'.$$

Proof: By induction on n ; it is easily seen that $\tilde{P}(k+1; k)$ consists of the single permutation

$$\sigma = (1) (k+1) (k) \dots (2),$$

which satisfies the conditions; and no other $\tau \in \tilde{S}_{k+1}$ satisfies these conditions.

Let $n > 1$, and suppose that the result is true for $n - 1$. Let $T \in \tilde{F}(nk+1, k)$, and label the vertices of T in pre-order. It follows from Definition 5.4 that the vertices labeled $(n-1)k + 2, \dots, nk + 1$ form a subtree rooted at the vertex w labeled $(n-1)k + 2$, and this sub-

tree is U_k . Let v be the parent of w ; then $label[v] \leq (n-1)k + 1$. Since a descendant of v is labeled $nk + 1$, it must be that the vertex labeled $(n-1)k + 1$ is a descendent of v ; furthermore, this vertex must equal v or lie in left subtree of v , since it can be seen that the right subtree of v is precisely the U_k already mentioned.

If the right subtree of v is deleted, the result is an element of $\tilde{F}((n-1)k+1, k)$ labeled in pre-order. By the induction hypothesis, the permutation generated from this tree is an element σ of $\tilde{P}((n-1)k+1; k)$. By the structural properties of T discussed above, it follows the permutation generated by T may be obtained from σ by inserting the sequence $(nk+1) (nk) \dots ((n-1)k+2)$ immediately to the right of $label[v]$ in σ ; but either $label[v] = (n-1)k + 1$ or $(n-1)k + 1$ lies to the left of $label[v]$ in σ ; hence the permutation generated by T satisfies (i), (ii), and (iii).

Let $\sigma \in \tilde{S}_{nk+1}$, $n \geq 2$, satisfy (i), (ii), (iii); let $T \in \tilde{B}(nk+1)$ be the binary tree generated from σ as in Theorem 4.4. By the induction hypothesis, the binary tree T' generated from $\sigma((n-1)k + 2, \dots, nk + 1)$ is an element of $\tilde{F}((n-1)k+1, k)$; also, the subtree of T corresponding to $(nk+1) (nk) \dots ((n-1)k + 2)$ is U_k ; so it remains to be shown that this U_k is the right subtree of some vertex v in T' . Let v be the parent of u , where $label[u] = (n-1)k + 2$. Suppose $u = lchild[v]$; now, $label[v] \leq (n-1)k + 1$, and v is a vertex in a subtree U_k of T' ; if v is not the leaf of U_k , it is obvious that $u = rchild[v]$; hence v is the leaf of the subtree U_k . Hence $label[v] = n'k + 1$ for some $n' < n$; also, $nk + 1$ occurs to the left of $n'k + 1$ in σ , from the construction of the inverse map in Theorem 4.4, but by (iii), $n'k + 1$ occurs to the left of $(n-1)k + 1$; hence $nk + 1$ occurs to the left of $(n-1)k + 1$, which contradicts the assumption about σ . It

follows that $T \in \underline{P}(nk+1, k)$.

q. e. d.

Lemma 5.9 leads to a lexicographic ranking scheme for $\underline{P}(nk+1, k)$, as will be seen shortly; at this point, however, there will be a digression in this exposition.

5.10 Definition: $[\underline{P}(nk+1, m; k), \underline{P}(nk+1, m; k)]$. Let $n \geq 0, k \geq 2$,

$1 \leq m \leq nk + 1$. $\underline{P}(nk+1, m; k) = \{\sigma \in \underline{P}(nk+1, k) : s_m = nk+1\}$; $\underline{P}(nk+1, m; k) = |\underline{P}(nk+1, m, k)|$.

The digression will be the evaluation of $\underline{P}(nk+1, m; k)$; in a manner similar to Lemma 4.16, this will lead to an enumeration of the set of full k -ary trees on $nk + 1$ vertices.

It is clear that Lemma 5.9 is equivalent to the proposition that for $n \geq 1$, $\underline{P}(nk+1, k)$ is obtained from $\underline{P}((n-1)k + 1; k)$ by direct insertion of the string $(nk+1) (nk) \dots ((n-1)k + 2)$ into all positions of σ such that 5.9 (i), (ii), and (iii) are satisfied, performing these insertions for all $\sigma \in \underline{P}((n-1)k + 1; k)$. It follows from this that

$$\underline{P}(1, m; k) = \begin{cases} 1 & m = 1 \\ 0 & m \geq 2; \end{cases} \quad (5.11)$$

and for $n \geq 1$

$$\underline{P}(nk+1, m; k) = \begin{cases} 0 & m \leq n \\ \sum_{\mu=n}^{m-1} \underline{P}((n-1)k + 1, \mu; k) & n + 1 \leq m \leq (n-1)k + 2 \\ 0 & (n-1)k + 3 \leq m \leq nk + 1. \end{cases} \quad (5.12)$$

Equation 5.12 is easily transformed to the following recursion,

if the convention $P(nk+1,0;k) = 0$ is adopted:

$$P(nk+1,m;k) = P(nk+1,m-1;k) + P((n-1)k+1,m-1;k). \quad (5.13)$$

The solution of Recursion 5.13 with boundary condition 5.11 will require the following lemma.

5.14 Lemma: The following are identities:

(i) For integers $a, b, > 0, \ell \geq 0$

$$\sum_{\lambda=0}^{\ell} \binom{a+\ell-\lambda-1}{\ell-\lambda} \binom{b+\lambda-1}{\lambda} = \binom{a+b+\ell-1}{\ell}$$

(ii) For integer p , real r, s, t ,

$$\sum_{\lambda \geq 0} \binom{r-t\lambda}{\lambda} \binom{s-t(p-\lambda)}{p-\lambda} \frac{r}{r-t\lambda} = \binom{r+s-tp}{p}$$

(iii) For integer k, ℓ, m ,

$$\sum_{\lambda=1}^{\ell+1} \frac{1}{\lambda(k-1) + 1} \binom{\lambda k}{\lambda} \binom{m-\lambda k}{\ell+1-\lambda} = \binom{m}{\ell}.$$

Proof: (i) and (ii) are equations 12.16 of [F] and 1.2.6.26 of [K], respectively. To obtain (iii) note that

$$\begin{aligned} \sum_{\lambda=1}^{\ell+1} \frac{1}{\lambda(k-1) + 1} \binom{\lambda k}{\lambda} \binom{m-\lambda k}{\ell+1-\lambda} &= \sum_{\lambda=1}^{\ell+1} \frac{1}{\lambda} \binom{\lambda k}{\lambda-1} \binom{m-\lambda k}{\ell+1-\lambda} \\ &= \sum_{\lambda=0}^{\ell} \frac{1}{\lambda+1} \binom{(\lambda+1)k}{\lambda} \binom{m-(\lambda+1)k}{\ell-\lambda} \end{aligned}$$

and the last summation is obtained from (ii) by substituting

$$t = -k;$$

$$r = k;$$

$$p = \ell;$$

$$s = m - (\ell+1)k.$$

q. e. d.

In particular, it should be noted that Equation 5.14 (ii) may be proven without reference to generating functions; see [K] or [GK].

5.15 Theorem: $P(nk+1, m; k)$ is evaluated as follows:

$$\begin{aligned}
 (i) \quad P(1, m; k) &= \begin{cases} 1 & m = 1 \\ 0 & \text{otherwise;} \end{cases} \\
 (ii) \quad P(k+1, m; k) &= \begin{cases} 1 & m = 2 \\ 0 & \text{otherwise.} \end{cases} \\
 (iii) \quad P(nk+1, m; k) &= \begin{cases} \binom{m-3}{n-2} & 1 \leq m \leq n+1 \\ \binom{m-3}{n-2} - \sum_{\lambda=1}^{\ell} \frac{1}{\lambda(k-1)+1} \binom{\lambda k}{\lambda} \binom{m-\lambda k-3}{n-\lambda-1} & \ell(k-1)+n+2 \leq m \leq (\ell+1)(k-1)+n+1 \\ \binom{(n-2)k}{n-2} - \sum_{\lambda=1}^{n-3} \frac{1}{\lambda(k-1)+3} \binom{\lambda k}{\lambda} \binom{(n-\lambda-2)k}{n-\lambda-1} & (n-2)k+4 \leq m \leq (n-1)k+2 \\ 0 & (n-1)k+3 \leq m \leq nk+1 \end{cases}
 \end{aligned}$$

Proof: The first two equalities are obvious; the proof of the case $n \geq 2$ is by induction on n .

For the case $n = 2$ it is easily seen by construction that

$$P(2k+1, m; k) = \begin{cases} 0 & 1 \leq m \leq 2 \\ 1 & 3 \leq m \leq k+2 \\ 0 & k+3 \leq m \leq 2k+1; \end{cases}$$

these equalities are in the form of (iii) given that there is no solution

to the inequalities on ℓ . For the case $n = 3$, the result is

$$P(3k+1, m; k) = \begin{cases} 0 & 1 \leq m \leq 3 \\ m - 3 & 4 \leq m \leq k + 3 \\ k & k + 4 \leq m \leq 2k + 2 \\ 0 & \end{cases}$$

which is also in the form of (iii) with $\ell = 0$. Assume that (iii) holds for $2, 3, \dots, n - 1$. It is easy to see directly that in the range $1 \leq m \leq n$, $P(nk+1, m; k) = 0$, and that $P(nk+1, n+1; k) = 1$. This gives the first equality of (iii).

In the second equality of (iii) the case $\ell = 0$ follows from the induction hypothesis on n , the first equality in (iii), Equation 5.13, and Pascal's identity on binomial coefficients. Assume that $\ell \geq 1$ and that the second equality of (iii) holds for $\ell - 1$. Let $m = \ell(k-1) + n + 2$; by the induction hypothesis on n ,

$$P((n-1)k+1, m-1; k) = \binom{m-4}{n-3} - \sum_{\lambda=0}^{\ell} \frac{1}{\lambda(k-1)+1} \binom{\lambda k}{\lambda} \binom{m-\lambda k-4}{n-\lambda-2};$$

by the hypothesis on ℓ ,

$$\begin{aligned} P(nk+1, m-1; k) &= \binom{m-4}{n-2} - \sum_{\lambda=0}^{\ell-1} \frac{1}{\lambda(k-1)+1} \binom{\lambda k}{\lambda} \binom{m-\lambda k-4}{n-\lambda-1} \\ &= \binom{m-4}{n-2} - \sum_{\lambda=0}^{\ell} \frac{1}{\lambda(k-1)+1} \binom{\lambda k}{\lambda} \binom{m-\lambda k-4}{n-\lambda-1}, \end{aligned}$$

since the second binomial coefficient in the $\lambda = \ell$ term of this summation is zero. By Equation 5.13 and Pascal's identity the result follows for $m = \ell(k-1) + n + 2$. For $\ell(k-1) + n + 2 < m \leq (\ell+1)(k-1) + n + 1$, the result follows by induction on m using Equation 5.13 and Pascal's identity. Hence the result holds for $n + 2 \leq m \leq (n-3)k + 4$ by induction on ℓ .

Suppose $(n-3)k + 5 \leq m \leq (n-2)k + 3$. Let $m' = (n-3)k + 4$,

$t = m - m'$. Then

$$\begin{aligned} P(nk+1, m; k) &= P(nk+1, m'; k) + t \left\{ \binom{(n-3)k}{m-3} - \sum_{\lambda=1}^{n-4} \frac{1}{\lambda(k-1)+1} \binom{\lambda k}{\lambda} \binom{(n-\lambda-3)k}{n-\lambda-2} \right\} \\ &= \binom{(n-3)k+1}{n-2} + \binom{(n-3)k}{n-3} \binom{t}{1} \\ &\quad - \sum_{\lambda=1}^{n-4} \frac{1}{\lambda(k-1)+1} \binom{\lambda k}{\lambda} \left\{ \binom{(n-\lambda-3)k+1}{n-\lambda-1} + \binom{(n-\lambda-3)k}{n-\lambda-2} \binom{t}{1} \right\}. \end{aligned}$$

From 5.14 (i) we have

$$\binom{(n-3)k+1}{n-2} + \binom{(n-3)k}{n-3} \binom{t}{1} = \binom{m-3}{n-2} - \sum_{v=2}^{n-2} \binom{(n-3)k+1-v}{n-2-v} \binom{t-1+v}{v};$$

similarly,

$$\binom{(n-\lambda-3)k+1}{n-\lambda-1} + \binom{(n-\lambda-3)k}{n-\lambda-2} \binom{t}{1} = \binom{m-\lambda k-3}{n-\lambda-1} - \sum_{v=2}^{n-\lambda-1} \binom{(n-\lambda-3)k+1-v}{n-\lambda-1-v} \binom{t-1+v}{v}.$$

Thus,

$$\begin{aligned} P(nk+1, m; k) &= \binom{m-3}{n-2} - \sum_{\lambda=1}^{n-4} \frac{1}{\lambda(k-1)+1} \binom{\lambda k}{\lambda} \binom{m-\lambda k-3}{n-\lambda-1} \\ &\quad - \sum_{v=2}^{n-2} \binom{(n-3)k+1-v}{n-2-v} \binom{t-1+v}{v} \\ &\quad + \sum_{\lambda=1}^{n-4} \sum_{v=2}^{n-\lambda-1} \frac{1}{\lambda(k-1)+1} \binom{\lambda k}{\lambda} \binom{(n-\lambda-3)k+1-v}{n-\lambda-1-v} \binom{t-1+v}{v}. \end{aligned} \tag{5.16}$$

Upon interchanging the order of summation, the last term becomes

$$\sum_{\lambda=1}^{n-4} \frac{1}{\lambda(k-1)+1} \binom{\lambda k}{\lambda} \binom{(n-\lambda-k)k-1}{n-\lambda-3} \binom{t+1}{2}$$

$$+ \sum_{v=3}^{n-2} \sum_{\lambda=1}^{n-1-v} \frac{1}{\lambda(k-1)+1} \binom{\lambda k}{\lambda} \binom{(n-\lambda-3)k+1-v}{n-\lambda-1-v} \binom{t-1+v}{v}.$$

Let $3 \leq v \leq n-2$, let $v + \mu = n-2$. Then the coefficient of $\binom{t-1+v}{v}$

in Equation 5.16 is

$$- \binom{(n-3)(k-1)+\mu}{\mu} + \sum_{\lambda=1}^{\mu+1} \frac{1}{\lambda(k-1)+1} \binom{\lambda k}{\lambda} \binom{(n-\lambda-3)(k-1)+\mu-\lambda}{\mu+1-\lambda} \\ = 0$$

by 5.14 (iii).

The coefficient of $\binom{t+1}{2}$ in Equation 5.16 is

$$- \binom{(n-3)k-1}{n-4} + \sum_{\lambda=1}^{n-4} \frac{1}{\lambda(k-1)+1} \binom{\lambda k}{\lambda} \binom{(n-\lambda-3)k-1}{n-\lambda-3} \\ = - \frac{1}{(n-3)k+1} \binom{(n-3)k}{n-3} \binom{-1}{0},$$

again, by Equation 5.14 (iii). Now

$$t+1 = m - m' + 1 \\ = m - (n-3)k - 3,$$

and by the standard convention for binomial coefficients

$$\binom{-1}{0} = 1.$$

Thus

$$P(nk+1, m; k) = \binom{m-3}{n-2} - \sum_{\lambda=1}^{n-3} \frac{1}{\lambda(k-1)+1} \binom{\lambda k}{\lambda} \binom{m-\lambda k-3}{n-\lambda-1},$$

and the result is established for $\ell = n-3$. For $(n-2)k+4 \leq m \leq (n-1)k+2$, $P((n-1)k+1, m-1; k) = 0$, by the induction hypothesis on n ; the evaluation of $P(nk+1, m; k)$ in this range follows from substituting $\ell = n-3$, $m = (n-2)k+3$ in the previous formula and applying Recursion 5.13. The last equality follows directly from the definition of

$P(nk+1;k)$.

q. e. d.

$$5.17 \text{ Corollary: } P(nk+1;k) = \frac{1}{n(k-1)+1} \binom{nk}{n}.$$

Proof: By the properties of $P(nk+1;k)$, $P(nk+1;k) = P((n+1)k,m;k)$ for $(n-1)k + 4 \leq m \leq nk + 2$. For m in this range

$$\begin{aligned} P((n+1)k,m;k) &= \binom{(n-1)k}{n-1} - \sum_{\lambda=1}^{n-2} \frac{1}{\lambda(k-1)+1} \binom{\lambda k}{\lambda} \binom{(n-\lambda-1)k}{n-\lambda} \\ &= \sum_{\lambda=n-1}^n \frac{1}{\lambda(k-1)+1} \binom{\lambda k}{\lambda} \binom{(n-\lambda-1)k}{n-\lambda} \end{aligned}$$

(by 5.14 (iii))

$$= \frac{1}{n(k-1)+1} \binom{nk}{n}$$

q. e. d.

Thus Corollary 5.17 evaluates the set of full k -ary tress on $nk + 1$ vertices, without reference to generating functions.

Returning to the problem of ranking the elements of $P(nk+1;k)$ in lexicographic order, the first step is to obtain a result analogous to Lemma 4.12. The direct insertion order on $P(nk+1;k)$ has already been informally defined while establishing Equations 5.11 and 5.12.

5.18 Lemma: On $P(nk+1;k)$, the lexicographic order and direct insertion order coincide.

Proof: By induction on n ; the result is trivial for $n = 0, 1$ since $P(1;k) = P(k+1;k) = 1$. Let $n > 1$ and suppose the result holds for $n - 1$. As in Lemma 4.12, it suffices to show that for $\sigma < \tau$ lexicographically consecutive in $P((n-1)k+1;k)$, σ' , the lexicographically largest element of $P(nk+1;k)$ formed from σ by direct insertion, is

lexicographically less than τ' , the lexicographically least element of $\tilde{P}(nk+1;k)$ formed from τ by direct insertion.

Let $\sigma = s_1 \dots s_{(n-1)k+1}$, $\tau = t_1, \dots, t_{(n-1)k+1}$.

By the induction hypothesis and the properties of $\tilde{P}((n-1)k+1;k)$ there is an $\ell > 1$ such that

$$s_i = t_i, \quad 1 \leq i \leq \ell;$$

$$s_{\ell+1} < t_{\ell+1}.$$

Suppose that $\tau' < \sigma'$; it follows that $\sigma' \in \tilde{P}(nk+1,m;k)$ for $m \leq \ell + 1$.

By the definition of direct insertion order on $\tilde{P}(nk+1;k)$, $s_{m-1} = (n-1)k+1$; hence $t_{m-1} = (n-1)k + 1$. But this is impossible since $\tilde{P}((n-1)k+1;k)$ is in direct insertion order.

q. e. d.

The final step in preparation for the ranking and unranking algorithms on $\tilde{P}(nk+1;k)$ is the establishment of a result analogous to Lemma 4.19. The result is somewhat surprising in that the coefficients are not particularly related to the $\tilde{P}(nk+1,m;k)$.

5.19 Definition: $[C(i,j,k)]$. Let $k \geq 2$, $j \geq 0$, $i \geq (j+1)k - 1$; let $n \geq \max(n+2,i)$. Let $\sigma = s_1 \dots s_{nk+1} \in \tilde{P}(nk+1;k)$ satisfy

$$s_{(n-\ell-1)k+2} = (n-\ell)k + 1 \quad 0 \leq \ell \leq j-1;$$

let

$$s_{nk-i+1} = (n-j)k + 1.$$

Let $\tau = t_1 \dots t_{nk+1}$ be obtained from $\sigma((n-j-1)k+2, \dots, (n-j)k+1)$ by inserting the substring $((n-j)k+1)((n-j)k) \dots (n-j-1)k+2$ so that $t_{(n-j-1)k+2} = (n-j)k + 1$.

Define

$C(i,j,k) = \text{lexrank}(\sigma) - \text{lexrank}(\tau)$, where *lexrank* is computed with respect to $P(kn+1;k)$. If $0 \leq i < (j+1)k - 1$, define

$$C(i,j,k) = 0.$$

Note that for $i = (j+1)k - 1$, $\sigma = \tau$ and hence $C(i,j,k) = 0$.

The next few results establish some important properties of the $C(i,j,k)$.

5.20 Lemma: If $i \geq k$, then $C(i,0,k) = i - k + 1$.

Proof: In this case,

$$\sigma = s_1 \dots s_{nk-i} (nk+1) (nk) \dots ((n-1)k+2) s_{nk-i+1+k} \dots s_{nk+1},$$

and

$$\tau = s_1 \dots s_{nk-i} s_{nk-i+1+k} \dots s_{nk+1} (nk+1) (nk) \dots ((n-1)k+2).$$

The result follows from the nature of the direct insertion procedure.

q. e. d.

5.21 Lemma: Let $j \geq 0$, $i \geq (j+1)k$;

then

$$C(i,j,k) = 1 + \sum_{v=0}^j C(i-j+v-1,v,k).$$

Proof: In this case,

$$\sigma = s_1 \dots s_{nk-i} ((n-j)k+1) \dots ((n-j-1)k+2) s_{nk-i+k+1} \dots s_{nk+1}.$$

Let

$$\sigma_0 = s_1 \dots s_{nk-i} s_{nk-i+k+1} \dots s_{(n-j)k+1} \in P((n-j-1)k+1,k);$$

form σ' from σ_0 by inserting the substring $((n-j)k+1) \dots ((n-j-1)k+2)$ immediately to the right of $s_{nk-i+k+1}$, and for $j-1 \geq \ell \geq 0$, inserting $((n-\ell)k+1) \dots ((n-\ell-1)k+2)$ immediately to the right of $((n-\ell-1)k+1)$.

Finally, let

$$\sigma'' = s_1 \dots s_{nk-i} s_{nk-i+k+1} ((n-j)k+1) \dots ((n-j-1)k+2) s_{nk-i+k+2} \dots s_{nk+1}.$$

Then

$$\begin{aligned} C(i,j,k) &= 1 + \text{lexrank}(\sigma') - \text{lexrank}(\tau) \\ &= 1 + (\text{lexrank}(\sigma') - \text{lexrank}(\sigma'')) \\ &\quad + (\text{lexrank}(\sigma'') - \text{lexrank}(\tau)). \end{aligned}$$

But the last term is by definition $C(i-1,j,k)$; and the first term is

$$\sum_{v=0}^{j-1} C(i-j+v-1, v, k).$$

q. e. d.

5.22 Corollary: Let $j \geq 1$, $i \geq (j+1)k$; then $C(i,j,k) = C(i-1,j-1,k) + C(i-1,j,k)$

Proof: By Lemma 5.22,

$$\begin{aligned} C(i,j,k) &= 1 + \sum_{v=0}^j c(i-j+v-1, v, k) \\ &= C(i-1,j,k) + 1 + \sum_{v=0}^{j-1} C(i-j+v-1, v, k) \\ &= C(i-1,j,k) + 1 + \sum_{v=0}^{j-1} C([i-1]-[j-1]+v-1, v, k) \\ &= C(i-1,j,k) + C(i-1,j-1,k). \end{aligned}$$

q. e. d.

The definition of $C(i,j,k)$ and the properties derived therefrom may be summarized in the following recursion:

$$C(i,j,k) = \begin{cases} 0 & 1 \leq i \leq j(k+1) - 1; \\ i - k + 1 & j = 0, \quad i \geq k; \\ C(i-1, j-1, k) + C(i-1, j, k) & \text{otherwise.} \end{cases} \quad (5.23)$$

Thus the $C(i,j,k)$ satisfy Pascal's identity, but with different boundary conditions.

5.24 Theorem: Let $j \geq 0$, $i \geq (j+1)k - 1$;

then

$$C(i,j,k) = \binom{i}{j+1} - (k-1) \binom{i}{j}.$$

Proof: By induction on i, j . The case $j = 0$, $i \geq k - 1$ is the left boundary condition of Recursion 5.23.

Suppose that $j > 0$ and that the result is true for $j - 1$.

Let $i = (j+1)k - 1$. Then

$$\begin{aligned} & \binom{(j+1)k-1}{j+1} - (k-1) \binom{(j+1)k-1}{j} \\ &= \binom{(j+1)k-1}{j+1} - (k-1) \frac{((j+1)k-1) \dots ((j+1)(k-1)+1)}{j!} \\ &= \binom{(j+1)k-1}{j+1} - \frac{((j+1)k-1) \dots ((j+1)(k-1))}{(j+1)!} \\ &= 0 \\ &= C((j+1)k-1, j, k). \end{aligned}$$

The rest follows from Pascal's identity.

q. e. d.

It is worth noting at this point that, although the $C(i,j,k)$ have been defined only for $k \geq 2$, if the result of Theorem 5.24 is extended to the case $k = 1$, then

$$C(i,j,1) = \binom{i}{j+1}, \quad j \geq 0, \quad i \geq j.$$

Thus the $C(i,j,k)$ extended to the case $k = 1$ are a generalization of ordinary binomial coefficients, at least over their common range of definition.

From their definition, the $C(i,j,k)$ will play the same role in lexicographic ranking algorithm for $P(kn+1;k)$ as did the $P(i,j)$ for lexicographic ranking of $P(n)$. In this algorithm, the $C(i,j,k)$ are stored in the array $ck[1:n*k, 0:n-2]$; as before the details of computing this array are omitted from the algorithm. The permutation $\sigma \in P(kn+1;k)$ is represented by a doubly-linked list, but the representation and list operations are suppressed. The permutation $\tau \in P(kn+1;k)$ is derived from σ as in Definition 5.19.

5.25 Algorithm: Input: $\sigma \in P(kn+1;k)$ represented as a doubly-linked list
Output: the lexicographic rank of σ in $P(kn+1;k)$.

begin

procedure *klexrank* (σ, n, ℓ);

begin

temp := 0;

for $j := \ell$ until $n - 2$ do

if $s_{(n-j-1)*k+2} = (n-j)*k + 1$ then

for $i := (j+1)*k$ until $n*k$ do

if $s_{n*k-i+1} = (n-j)*k$ then

begin

temp := $ck[i, j, k] + klexrank(\tau, n, j+1)$;

i := $n*k$;

j := $n-2$;

end;

```

    klexrank: = temp;
end lexrank (σ, n, ℓ);
compute ck[1:n*k; 0:n-2];
lexrank (σ, n, 0);
end.

```

The $C(i, j, k)$ also lead to an unranking algorithm.

5.26 Algorithm: Input: integers $n \geq 0$, $0 \leq r \leq P(nk+1; k)-1$. Output: the rank r element of $P(nk+1; k)$ represented as a linked list.

```

begin
  procedure klexrecover (r, n, j);
  begin
    if j > 0 then
      begin
        find largest i such that ck[i, j] ≤ r;

        σ := s1s2...sn*k-i((n-j)*k+1)...((n-j-1)*k+2)sn*k-i+1...s(n-j-1)*k+1;

        klexrecover (r-ck[i, j], n, j-1);
      end;
    end klexrecover (r, n, j);
    compute ck[1:n*k, 0:n-2];
    σ := 1(k+1) (k) ... 2;
    klexrecover (r, n, n-2);
  end.

```

5.27 Theorem: Algorithms 5.25 and 5.26 are $O((nk)^2)$ time-bounded and $O(nk)$ space-bounded.

Proof: Similar to proofs of Theorems 4.21 and 4.24.

§VI Final Remarks: The notion of ranking combinatorial configurations was first presented in [L]; since that time, a considerable amount of work has been done on the subject; see [CW], [FW], [W1], [W2], [W3].

The ranking algorithms in §IV and §V differ from the usual strategy for ranking permutations and n -tuples. This strategy, a special case of the general strategy defined in [W2], may be described as follows: to rank $s_1 \dots s_n$ lexicographically, let $A_i = \{t_1 \dots t_n : t_j = s_j, 1 \leq j \leq i-1 \text{ and } t_i < s_i\}$. Then the rank of $s_1 \dots s_n$ is simply

$$\sum_{i=1}^n |A_i|.$$

If a linear order is defined on $\tilde{P}(n)$ as follows:

$\sigma < \tau$ in $\tilde{P}(n)$ iff

(i) $\sigma \in \tilde{P}(n, m)$, $\tau \in \tilde{P}(n, m')$ and $m < m'$; or

(ii) $\sigma, \tau \in \tilde{P}(n, m)$ and $\sigma(n) < \tau(n)$ in $\tilde{P}(n-1)$,

then the general strategy of [W2] may be applied to produce ranking and unranking algorithms; these algorithms are $O(n^2)$ time and $O(n)$ space-bounded. A similar remark applies to $\tilde{P}(nk+1; k)$.

REFERENCES

- [A] A. V. Aho, J. E. Hopcroft, J. D. Ullman The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, MA, 1973.
- [CW] E. Calabi and H. Wilf, On the Sequential and Random Selection of Subspaces Over a Finite Field, J. Combinatorial Theory, to appear.
- [F] W. Feller, An Introduction to Probability Theory and Its Applications, Volume I, Wiley, New York, 1957.
- [FW] J. Fillmore and S. G. Williamson, Ranking Algorithms: The Symmetries and Colorations of the n-Cube, SIAM Journal on Computing 5,2 (June 1976), pp. 297-304.
- [GK] H. W. Gould and J. Kaucky; Evaluation of a Class of Binomial Coefficient Summations, J. Combinatorial Theory 1 (1966), pp. 233-247.
- [K] D. E. Knuth, The Art of Computer Programming, Volume I, Addison-Wesley, Reading, MA, 1973.
- [L] D. H. Lehmer, The Machine Tools of Combinatorics, in Applied Combinatorial Mathematics, E. F. Beckenbach (ed.), Wiley, New York, 1964, pp. 5-31.
- [R] Rudin, W., Real and Complex Analysis, McGraw-Hill, New York, 1966.
- [W1] H. S. Wilf, A Unified Setting for Sequencing, Ranking, and Selection Algorithms for Combinatorial Objects, preprint, University of Pennsylvania.
- [W2] S. G. Williamson, On the Ordering, Ranking, and Random Generation of Basic Combinatorial Sets, in Proceedings of the Table Rondé, Combinatoire et Représentation du Groupe Symétrique, Strasbourg, France, 26-30 April, 1976, to appear.
- [W3] S. G. Williamson, Ranking Algorithms for Lists of Partitions, SIAM Journal on Computing, 5,4 (December 1976), pp. 602-617.

1. Report No. UIUCDCS-R-77-850		2.		3. Recipient's Accession No.	
4. Title and Subtitle On The Ordering, Enumeration and Ranking of k-ary Trees				5. Report Date February, 1977	
6.				7. Performing Organization Rept. No.	
8. Author(s) Anthony E. Trojanowski				9. Performing Organization Name and Address Department of Computer Science University of Illinois at Urbana-Champaign Urbana, IL 61801	
10. Sponsoring Organization Name and Address National Science Foundation Washington, DC				11. Project/Task/Work Unit No.	
				12. Contract/Grant No. MCS-73-03408	
				13. Type of Report & Period Covered	
				14.	
15. Supplementary Notes					
16. Abstracts The problem of ranking a finite set X may be defined as follows: if $ X = N$, define a linear order on X and find an order isomorphism $\varphi: X \rightarrow \{0, 1, \dots, N-1\}$. In this paper, X is the set of k -ary trees on n vertices, $k \geq 2$, $n \geq 0$. In addition to constructing ranking functions with respect to two different linear orders on k -ary trees, an investigation into some purely enumerative aspects of the problem is included. One result of this investigation is a generalization of binomial coefficients.					
17. Key Words and Document Analysis. 17a. Descriptors algorithm, binary tree, linear order, k-ary tree, ranking function, recursion					
18. Identifiers Open-Ended Terms					
19. OSATI Field/Group					
20. Availability Statement				21. Security Class (This Report) UNCLASSIFIED	
				22. Security Class (This Page) UNCLASSIFIED	
				23. No. of Pages	
				24. Price	

APR 1 4 1977



CITY OF ILLINOIS-URBAN
IL6R no. C002 no. 848 851(1977
Level restricted NOR network transductio



3 0112 088403305